A QoS architecture for MANETs supporting real-time peer-to-peer multimedia applications [*]

Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni, and Manuel P. Malumbres
Department of Computer Engineering
Polytechnic University of Valencia (UPV), Valencia, Spain
Phone: +34 96 387 7007
{calafate, jucano, pmanzoni, mperez}@disca.upv.es

Abstract

In this work we propose a QoS architecture for MANETs based on distributed admission control and the IEEE 802.11e technology. Our aim is to improve peer-to-peer communication in wireless mobile ad hoc networks by supporting real-time multimedia streaming. This technology can adapt to applications with bandwidth, delay and jitter constraints, and yet we keep to a minimum the requirements imposed on intermediate stations.

Simulation results show that we successfully achieve our goal of supporting QoS-constrained applications in MANETs with a low overhead, confirming the adequateness of using probe-based admission control in these environments.

1 Introduction

Real-time multimedia communication between peers is characterized by imposing QoS constraints on the underlying networks. Examples of such constraints are minimum bandwidth, maximum delay and jitter requirements.

The proliferation of devices with wireless capabilities in the last few years has increased the connectivity to the Internet; it has also opened new possibilities in the field of peer-to-peer communication, leading to the appearance of mobile ad hoc networks, also known as MANETs. MANETs consist of several independent mobile stations that cooperate with each other to create a network without requiring the presence of any sort of infrastructure. In MANETs all stations must be constantly adapting to varying link quality towards their neighbors, and also to changes in network topology (routing related adaptation). In such an environment it is extremely hard to achieve reliable communication between peers, and the situation becomes even more complex if some of the sources generate traffic with QoS requirements. However, the ratification of the IEEE 802.11e standard [6] (expected in the last quarter of 2005) will offer a significant performance boost to MANETs, making them more suitable to handle QoS traffic.

In the past there have been some proposals aiming at setting a framework for QoS support in MANET environments. Examples of such proposals are FQMM [5], INSIGNIA [10] and SWAN [3].

FQMM [5] has been presented as a flexible QoS model for MANETs. It proposes a hybrid per-flow and per-class provisioning scheme, so that traffic of the highest priority benefits from per-flow provisioning, while other category classes are given per-class provisioning; the IEEE802.11 MAC layer is used without changes. In a later work [4], though, authors find that the priority buffer and scheduling schemes proposed fail when UDP traffic gets higher priority than TCP.

Lee et al. [10] proposed INSIGNIA, an in-band signaling system that supports fast reservation, restoration and adaptation algorithms. With INSIGNIA all flows require admission control, resource reservation and maintenance at all intermediate stations between source and destination to provide end-to-end quality of service support. However, Georgiadis et al. [9] show that link interferences (due to the hidden terminal problem) in multihop wireless networks make the problem of selecting a path satisfying bandwidth requirements an NP-complete problem, even under simplified rules for bandwidth reservation.

Ahn et al. [3] designed SWAN, a stateless network model aiming at providing service differentiation in MANETs. One of the main advantages of SWAN is that it does not require the support of a QoS-capable MAC to provide service differentiation; instead, it uses plain IEEE 802.11. SWAN's admission control mechanism requires all stations to keep track of the MAC's transmission delay of all packets in order to estimate available bandwidth; however, the associa-

[*]This work has been partially supported by the Generalitat Valenciana under grant GV05/245.

tion of a global estimate for transmission delay with a certain bandwidth in the link towards a specific station is not straightforward, especially outside simulation scope.

In this work we propose a solution which we named DACME: Distributed Admission Control for Manet Environments. DACME offers a new framework for QoS support in MANETs based on the IEEE 802.11e technology. With our work we expect to increase performance and reduce the requirements on MANET terminals relatively to previous proposals. The purpose of DACME is offering a distributed admission control mechanism to support real-time peer-to-peer multimedia applications for MANETs. Due to the probe-based nature of DACME, its implementation and deployment in real-life MANETs is effective and yet simple.

The rest of this paper is organized as follows: in section 2 we expose the core of our proposal (DACME). In section 3 we present some performance results and finally, in section 4, we present our conclusions as well as future work.

2 The distributed admission control mechanism

DACME is a probe-based admission control mechanism that performs end-to-end QoS measurements according to the QoS requirements of multimedia streams. In order for DACME to operate in optimal conditions in IEEE 802.11-based MANETs, all the radio interfaces should be IEEE 802.11e enabled. However this is not a strict requirement, which means that DACME will still operate correctly in non QoS-compliant MANETs.

In terms of software restrictions on MANET nodes, only the source and destination of a QoS flow must have a DACME agent running. The rest of the nodes will simply treat DACME packets as regular data packets, being unaware of the mechanism itself.

Concerning DACME's components, in figure 1 we present the functional block diagram of a DACME agent. The main element of DACME is the *QoS measurement module*. This module is responsible for assessing QoS parameters on an end-to-end path. Another important element is the *Packet Filter*. Its purpose consists in blocking all traffic which is not accepted into the MANET, and altering the IP TOS (Type of Service) packet header field in the packets of all accepted flows according to the QoS that has been requested.

An application that wishes to benefit from DACME must register with the DACME agent by indicating the source and destination port numbers, the destination IP address and the required QoS parameters; these data are stored internally in a table indexed using source port numbers.

Once registration is completed successfully the *QoS measurement module* is activated and will periodically per-

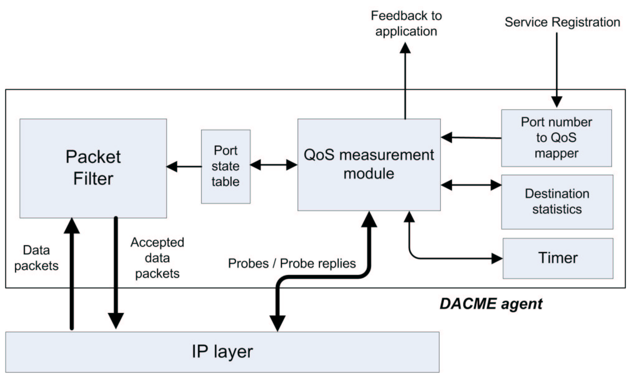


Figure 1. Functional block diagram of the DACME agent

form path probing between source and destination in order to assess the current state of the path in terms of available bandwidth, end-to-end delay and jitter. The destination agent, upon receiving probe packets, will update the *Destination statistics* table where it keeps per source information of the packets received during the current probe. After receiving the last packet of a probe (or if a timeout is triggered) the destination agent will send a reply back to the source DACME agent. The *QoS measurement module*, upon receiving each probe reply, will update the state of the path accordingly. Once enough information is gathered it checks all the registered connections towards that destination, and then decides when a connection should be accepted, preserved or rejected, updating the *Port state table* accordingly (with either accept or drop). If only part of the registered connections can be allowed, preference is given to those which have registered first. This module can also notify applications about service events using a feedback call, if requested at service registration.

Actual QoS support is achieved when the *Packet Filter* element, according to the data in the *Port state table*, configures the IP TOS (Type of Service) packet header field on packets belonging to accepted data flows, according to the requested QoS. The IEEE 802.11e MAC must then map the service type defined in the IP TOS packet header field to one of the four MAC Access Categories available: Voice, Video, Best effort and Background.

In a previous work [1] we studied which is the most appropriate technique and the number of packets to be used for each probe type, without actually implementing the admission control mechanism itself. In this paper we evaluate an actual implementation of DACME based on the findings of that earlier work.

In the next three sections we will detail DACME's mechanisms used to determine end-to-end bandwidth, delay and jitter.

Algorithm 1 Strategy followed to reach bandwidth verdicts

```
After receiving a bandwidth probe reply do {
 re-calculate the corrected bandwidth estimation value using all available measurements
 if (there is a level of confidence of 95% that the available bandwidth is higher that the requested one)
   then set the bandwidth flag to true
 else if (there is a level of confidence of 95% that the available bandwidth is lower that the requested one)
   then set the bandwidth flag to false
 else if (number of probes used is less than maximum allowed)
   then send a new probe
 else maintain the previous bandwidth flag value }
```

2.1 Support for bandwidth constrained applications

The support for bandwidth constrained applications is achieved using periodic end-to-end measurements of available bandwidth using probes. According to the results found in [1], bandwidth probes consist of ten back-to-back packets sent to the destination using UDP.

The destination's DACME agent will use the probe to obtain a measure of the average throughput received ($B_{measured}$), sending a probe reply back to the source. The source's DACME agent, when receiving the probe reply packet, will collect the $B_{measured}$ values sent by the destination agent to be able to reach a decision on whether to admit the connection or not.

We always configure bandwidth probes so that probe packets are mapped to the *Video* MAC Access Category independently of the type of service registered by the application. This way we avoid that a higher priority connection (e.g. voice) causes the degradation of an on-going connection with lower priority (e.g. video) if both connections are generated by the same user, therefore sharing the same terminal; this interaction among traffic of different priorities is also known as the *stolen bandwidth problem* [8].

We now propose a probabilistic strategy to reach a bandwidth-related verdict according to probe measurements. This strategy is the one described in algorithm 1.

This algorithm allows reducing the number of probes required to take a decision to a value as low as two probes; such a fast decision occurs often in those situations where it becomes quickly evident that the available bandwidth is either much higher or much lower than the requested one. The maximum number of probes allowed per cycle is set to five, also according to the analysis performed in [1].

In terms of applications with bandwidth constraints requesting DACME's services, their traffic is only accepted if a positive verdict is reached based on bandwidth measurements (bandwidth flag is set to true).

Since MANETs are an environment prone to frequent packet losses, both source and destination must accommodate to this event. Relatively to the DACME agent at the source, it keeps a timer to be able to react in case a probe reply is never received. This timer is set to go off 500 ms after sending the probe. If no probe reply is received, causing the timer to be triggered, or in the case that the probing process is completed successfully, the source will schedule a new probing cycle after 3 seconds ± 500ms of jitter to avoid possible negative effects due to probe synchronization. This value was carefully chosen taking into account the typical topology change rates, and intends to offer a balance between the performance drop caused by poor reaction times and the performance penalty introduced by the probing process itself.

Concerning the DACME agent in the destination, it must also accommodate to the event that, when receiving a probe, not all packets arrive. So, it must keep a timer to avoid delaying the reply to the source too much, thereby increasing the responsiveness. If the timer goes up and the destination did not receive enough probe packets (al least half of the packets sent), it notifies such occurrence to the source (null probe).

In the next section we extend this technique to support delay constraints also.

2.2 Support for delay-constrained applications

When an application has bandwidth and also delay requirements, the DACME agent is required to extend the process described before to handle this new constraint. Here we assume that applications with delay constraints are also bandwidth constrained, though in our implementation of DACME we also support applications with delay constraints only.

The technique used to measure end-to-end delay as part of DACME's architecture is similar to the measurements made by a ping application, dividing by two the round-trip time obtained. Relatively to *ping*, the main difference in DACME is that a new *echo request* packet is sent immediately after receiving an *echo reply* packet to reduce as much as possible the time used to perform measurements. Also, the *echo reply* packet should have the same length and the same IP TOS field as the *echo request* one. The value of the IP TOS field in end-to-end delay probes is the same one requested by the application.

According to the previous analysis [1] we require at least three consecutive round-trip times to obtain a reliable value. Therefore, the technique we use to handle applications with both delay and bandwidth requirements is the following: we start with four consecutive rounds of *ping/pong delay probes* to assess the end-to-end delay. The value of the first round is discarded since it is used as a worm-up round to trigger routing and find end-to-end bidirectional paths if

Algorithm 2 Strategy followed to reach delay verdicts

```
After executing code from algorithm 1 do {
  if (traffic is currently blocked)
    then find worst and best case estimates for delay using expression 1
  else use the measured delay as the best and worst case delay
  if (best case delay $>$ maximum delay allowed)
    then set delay flag to false
  else if (worst case delay $<$ 90% of the maximum delay allowed)
    then set delay flag to true
  else if (number of bandwidth probes used $<$ maximum allowed)
    then send a new probe
  else maintain the previous delay flag value }
```

necessary. The results from the remaining three rounds are averaged and stored.

In a similar way to what we have done for bandwidth, we have to correct the delay estimation in case that the traffic is currently blocked. The estimator used is the one presented in (3) according to the results found in [1].

$$D_{e3}(x) = e^{\alpha \cdot x + \beta} + \gamma + \eta \cdot x + \vartheta \cdot x^2 \qquad (1)$$

In this expression x is the normalized bandwidth value and $(\alpha, \beta, \gamma, \eta, \vartheta)$ are experimental parameters; the optimum value for them was found using least-squares regression. The value of x is obtained by the ratio between the application's chosen data rate and the bandwidth estimated through the probing process. Relatively to D_e values, these are normalized delay values obtained dividing end-to-end delay values by the low-rate delay estimation (obtained by the ping-pong probing process).

Using estimator D_{e3} we can obtain worst and best case estimates for the end-to-end delay by using the worst and best case estimates for available bandwidth respectively. As a final step we must multiply the normalized delay values by the low-rate value obtained in the delay probing process to de-normalize it.

The strategy followed in this algorithm consists in rectifying end-to-end delay by finding worst and best case estimations in case the traffic is blocked. When traffic is flowing there is no need to perform adjustments, and the mean of measured values is directly used.

For those applications requesting DACME's services with both bandwidth and delay constraints, their traffic is only accepted when a positive verdict is reached in terms of both bandwidth and delay. In the next section we extend this technique to support jitter constraints.

2.3 Support for jitter-constrained applications

In this section we complete DACME's QoS framework by including support for jitter-constrained applications. We couple jitter with bandwidth measurement events, measuring jitter after bandwidth probing ends and only when necessary, as explained next.

Algorithm 3 Strategy followed to reach jitter verdicts

```
After obtaining jitter measurements do {
  if (2 $\times$ standard deviation $<$ maximum jitter)
    then set jitter flag to true
  else if (1.9 $\times$ standard deviation $>$ maximum jitter)
    then set delay flag to false
  else maintain the previous jitter flag value }
```

Relatively to the jitter measurement process, the source must send packets with the size, IP TOS field value and data rate chosen by the application for 250 ms according to [1]. The receiving end, aware of the source's packet sending rate by explicit notification, calculates the mean and standard deviation values for the absolute jitter and returns them to the source.

These measurements are only performed if the application's traffic is blocked, and they are only performed after delay and bandwidth probes if both tests return a positive verdict. In case that the traffic from the application is already flowing through the network there is no need to send jitter probes. In that situation the destination agent can measure the jitter of the actual traffic received and send it back to the source.

Independently of the method used to measure jitter (probes or actual traffic), once the source receives jitter statistics (which consist of absolute mean and standard deviation values) it will assess compliance with the maximum value requested by the application through algorithm 3.

Since jitter follows a normal distribution with a mean value of zero, about 95% of the cases fall between $\pm 2\sigma$. Therefore, in our algorithm we accept traffic only if 95% of the packets have a jitter value lower than the maximum requested. We also introduce hysteresis by defining a zone between 1.9 and 2σ where the strategy consists of maintaining the previous flag value to reduce traffic fluctuations.

To conclude, if an application imposes bandwidth, delay and jitter constraints to the DACME agent, its traffic is only accepted when a positive verdict is reached in terms of all bandwidth, delay and jitter.

3 DACME performance in MANETs

In this section we evaluate the effectiveness of our QoS architecture using simulation. Experiments are conducted using the ns-2 [7] discrete event simulator. All simulations are conducted in a typical MANET environment sized 1900x400 squared meters with 50 nodes. The choice of the scenario aims simultaneously at avoiding network partitioning and increasing the average number of hops. Nodes are moving at a constant speed of 5 m/s according to the random waypoint mobility model. Concerning the nodes' radio interfaces, these are IEEE 802.11g/e enabled. We changed the physical layer of ns-2 nodes to make it compliant with

IEEE 802.11g, and we use the IEEE 802.11e extentions by Wietholter and Hoene [11].

Relatively to the radio range, it is of 250 meters, leading to an average of 4 hops between nodes. With this setting we consider that the routing protocols are conveniently stressed, causing a significative number of path changes throughout simulations.

Concerning traffic, we divide it in two groups: background traffic and data traffic. Background traffic is used to obtain different congestion levels in the network, while data traffic is regulated by DACME and is used to assess DACME's effectiveness.

We have four sources of background traffic which generate negative-exponentially distributed traffic in the *Video*, *Best Effort* and *Background* Access Categories of the MAC layer. The traffic share for each Access Category is of 50% to the Video AC and 25% to both Best Effort and Background ACs. We do not generate background traffic for the Voice Access Category because it was designed to support low data-rate streams such as voice streams; moreover, we want to avoid provoking routing misbehavior since routing traffic is also set to the Voice AC.

Concerning the data sources under study (regulated by DACME), these consist of four video streams and three voice streams. The video sources send CBR traffic at 1 Mbit/s using 512 byte packets. Voice sources are VoIP streams simulated using a Pareto On/Off distribution with both burst and idle time set to 500 ms. The shaping factor used is 1.5 and the average data rate is of 100 kbit/s. In order to adjust to bandwidth fluctuations, and to reserve some extra bandwidth for routing tasks, we need to set the minimum amount of additional bandwidth that must be reserved; for the routing protocol we chose for study (AODV [2]) we found through simulation that this extra bandwidth should be of 0.75 Mbit/s to achieve optimum performance. Relatively to start and end times for the different sources, the first video source is started at the beginning of the simulation, and then every 15 seconds a new data source becomes active, alternating between voice and video sources. Each source is active for two minutes, and all results presented are average values over ten simulation runs, each lasting 400 seconds.

3.1 Mobility and routing issues

In section 1 we described an admission control architecture that is independent of the routing protocol used. However, we consider that the DACME agent, and therefore the application that relies on it, can benefit from obtaining awareness of the state of a path as seen by the routing protocol. Therefore, we improved the implementation of DACME so that the packets that are passed between the IP layer and the QoS measurement module (see figure 1) are not only DACME probe/reply packets, but also routing packets. The routing protocol we chose for study was AODV [2]. AODV is a reactive routing protocol that only performs routing tasks when there is actual traffic requiring it. Basically, when a route must be found a RREQ packet is propagated through broadcasting throughout the MANET until the destination is reached. The destination will then send a RREP packet back to the source and communication can be started. In case the break of a link is detected, the node detecting the failure sends a RERR packet to the source that must start a new route discovery cycle.

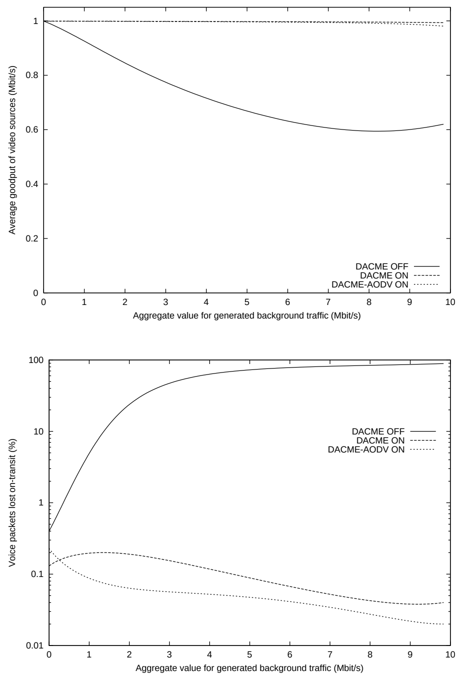


Figure 2. Improvements in terms of video goodput (top) and voice packet losses (bottom) by using DACME and DACME-AODV

Relatively to the integration with DACME, we consider that the DACME agent could improve performance if it acts as soon as a new path is established (after receiving a RREP packet) so as to assess if it can sustain the desired QoS. The integration of DACME with AODV proposed in this section will be referred to as DACME-AODV from now on.

3.2 DACME's performance supporting applications with bandwidth requirements

Supporting applications with QoS constraints implies assuring minimum bandwidth requirements, and sometimes also delay and jitter requirements.

In this section we focus on DACME's effectiveness in supporting bandwidth-constrained multimedia streams. We achieve our purpose by varying the amount of traffic generated by background sources and observe the performance

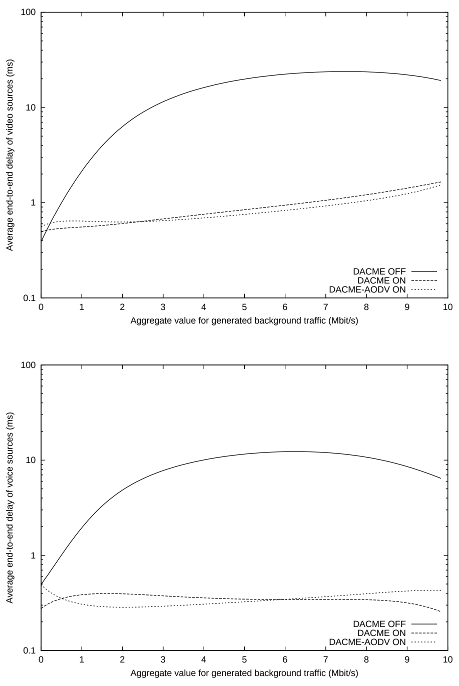


Figure 3. Average end-to-end delay values for video (top) and voice (bottom) sources.

experienced by the voice and video streams under focus.

Figure 2 shows the improvements in terms of video goodput and voice packet losses using DACME and DACME-AODV compared to a solution where DACME is not used (turned off). We can observe that when DACME is not used the average goodput for the different video sources drops steadily with increasing congestion. By using DACME the average goodput is maintained close to maximum because sources are only allowed to transmit if the DACME agent verifies that the available bandwidth is enough. Obviously, as the congestion level increases, the amount of DACME-regulated traffic accepted into the network must decrease. We find that, under low background congestion, the traffic acceptance rate for DACME regulated sources is about 90% for Voice and 93% for Video for both DACME versions; under very high congestion the traffic acceptance rate decreases to about 75% for Voice traffic and to 35% for Video traffic.

From figure 2 we can also observe that DACME offers great improvements in terms of voice packet losses. We see that, if DACME is not used, the increase in congestion will cause voice packet losses to achieve extremely high values; as expected, DACME-AODV offers a better performance than the default DACME implementation.

We now proceed to evaluate the performance achieved in terms of end-to-end delay. In figure 3 we see that when using DACME the end-to-end delay for both Voice and Video sources is greatly improved. Comparing DACME with DACME-AODV the differences in terms of average values are only slight.

An interesting way to gain further insight on the benefits of DACME in MANET environments is to analyze the stability in terms routing overhead, or the lack of it. In this work we found that DACME, by regulating congestion, avoids routing collapse situations. In fact we verify that, when DACME is not used, the routing overhead can be increased up to four times more.

Relatively to DACME's overhead, we observe that it is maintained at low levels: the average overhead is about 36 kbit/s for DACME and about 43 kbit/s for DACME-AODV, both never reaching 50 kbit/s. This is a very reasonable value taking into consideration that we are following a probe-based approach.

It is interesting to notice that, as congestion increases, the amount of video traffic admitted decreases at a steady rate, contrarily to voice traffic. This is due to the fact that video streams require a much larger bandwidth share.

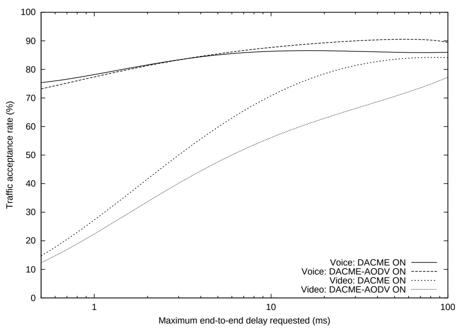
3.3 DACME's performance supporting applications with bandwidth and delay requirements

In this section we will assess the effectiveness of DACME in supporting delay-bounded applications. With that aim we choose a fixed value for aggregate background traffic to proceed with our experiments. The chosen value is of 2.3 Mbit/s, and it is maintained throughout the simulations.

The simulations made are similar to those performed for bandwidth-constrained applications. The only difference is that we now notify the DACME agent that the applications are also delay bounded, setting different values for maximum end-to-end delay. In our experiments these values vary between 0.1 and 100 ms.

When analyzing the simulation results we observed that when applying a maximum delay threshold of 0.1 ms no video or voice traffic was accepted into the network (cut-off value). In figure 4 we present the traffic acceptance curves when varying the maximum delay settings. We observe that the impact of imposing delay requirements is more pronounced on video sources, being that the voice traffic only varies slightly; as expected, when demanding relatively high values for end-to-end delay (100 ms) the amount of traffic accepted for both video and voice sources is close to the one found when applying bandwidth-constraints only. Also, for video traffic, we can observe that DACME-AODV gets lower traffic acceptance rates, which is mainly due to earlier reaction to route changes that causes connections to be blocked more often at the specified network load.

If we now take into consideration the percentage of traffic that meets the requested maximum value for end-to-end delay, we observe that voice traffic meets the requirements more strictly than video traffic (see figure 4). These results



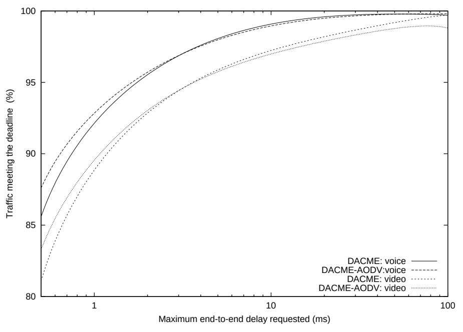


Figure 4. Traffic acceptance rate values (top) and percentage of traffic meeting the end-to-end delay deadline (bottom) for video and voice traffic when varying the maximum end-to-end delay requested

also allow measuring the effectiveness of the DACME architecture in complying with the end-to-end requirements made. We find that, although DACME agents only re-assess the end-to-end delay values every 1.5 seconds when traffic is flowing, this strategy offers good results even when the scenario is characterized by an important degree of mobility: more than 80% of the accepted traffic meets the deadline always. We also find that DACME-AODV offers better results when the maximum delay requested is very low.

To conclude the evaluation, we now analyze the average overhead per source introduced by DACME. In the previous section we found that, at the selected degree of congestion (aggregated background traffic value equal to 2.3 Mbit/s), DACME's overhead was found to be around 37 and 43 kbit/s for DACME and DACME-AODV respectively. We verify that by introducing additional probes to measure end-to-end delay does not have a significant impact on overhead. In fact we find that, when the requested end-to-end delay is low, DACME's overhead is inferior to the one found without delay constraints (around 35 kbit/s). This occurs because sometimes the delay requirements allow reaching a *deny* verdict right from the start, obviating the need to reach a verdict for bandwidth .

Once we reach relatively high values for requested end-to-end delay we find that the overhead compared to the bandwidth-constrained solution is increased by 14 and 17 kbit/s respectively, quite acceptable values.

We will now proceed by doing a similar analysis in the scope of jitter bounded applications.

3.4 DACME's performance supporting applications with bandwidth, delay and jitter requirements

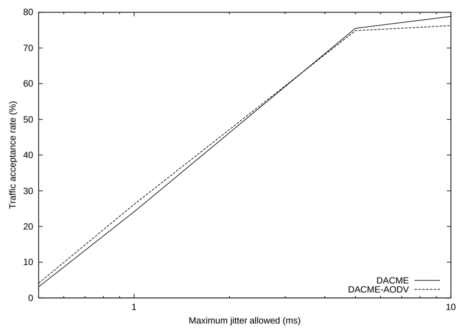
In the previous section we observed the effectiveness of DACME to support applications with both bandwidth and end-to-end delay constraints. In this section we take a final step to evaluate the effectiveness of DACME in supporting applications with bandwidth, delay and also jitter requirements. With this purpose we maintain all the simulation parameters used in the previous section, fixing the value for the maximum end-to-end delay requested at 10 ms. We now impose different requirements for jitter, with values ranging from 0.1 ms up to a maximum value of 10 ms.

Relatively to traffic, we have video sources generating CBR traffic and voice sources generating traffic according to a Pareto on-off distribution. Since, from the destination point of view, it is only meaningful to assess the jitter of traffic with a constant packet arrival rate, the results presented in this section refer only to video traffic; voice sources are also included as before, but they only have bandwidth and delay constraints.

In figure 5 we show the variation in terms of accepted video traffic as the maximum jitter allowed increases. We observe that 0.1 ms is a cut-off value for jitter, and that when the maximum jitter allowed is of 10 ms the traffic acceptance rate is similar to the one found without jitter constraints.

We now proceed by analyzing the amount of video traffic that meets the jitter requirements imposed. The results of figure 5 show that, when the jitter limits are too low, only about 2/3 of the traffic meets these limits. As we relax the jitter constraints the percentage of traffic meeting the requirements increases significantly. It is interesting to notice that, despite the differences between DACME and DACME-AODV found in figure 5, in terms of percentage of traffic meeting jitter requirements the difference is not so substantial, though plain DACME is still slightly superior to DACME-AODV.

To conclude this section we now study the overhead of DACME when jitter probes are also included. In the previous section we found that DACME and DACME-AODV generate and overhead of 47 and 45 kbit/s respectively for a maximum end-to-end delay of 10 ms. Adding jitter constraints has increased these values to 76 and 90 kbit/s respectively for the best case, which is a significative but tolerable increase. On worst case situations the maximum overhead for DACME is below 120 kbit/s.



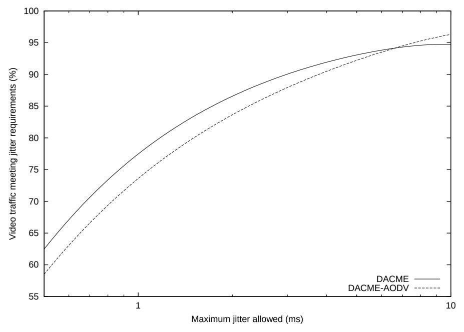


Figure 5. Traffic acceptance rate growth (top) and percentage of traffic meeting the maximum jitter value requested (bottom) by increasing the maximum jitter allowed

4 Conclusions and future work

In this paper we presented a novel QoS architecture which enables real-time multimedia communication between peers in wireless mobile ad hoc networks. The core of our architecture consists of DACME, a probe-based distributed admission control mechanism capable of supporting bandwidth, delay and jitter QoS requirements. Our proposal can be easily deployed since it imposes very few requirements on the stations participating in the MANET. In fact, these stations only need to have IEEE 802.11e capable interfaces and to handle packets according to the TOS field in their IP header. This relaxation of the requirements on MANET stations is an important improvement over previous proposals.

Simulation results show that the probabilistic admission control technique used in DACME is effective at different levels of congestion, and that delay and jitter constraints are met with a good level of accuracy. Overall, DACME improves the performance experienced by users and also avoids wasting MANET resources. We observe that enhancing DACME with routing awareness further improves the performance achieved. Relatively to the overhead introduced by DACME's mechanism, we found it to be quite low: between 30 and 60 kbit/s, except when jitter support is also required, in which case it can reach values up to 120 kbit/s.

As future work we plan to enhance DACME to support scalable video flows, as well as to develop an implementation of DACME for GNU/Linux platforms.

References

[1] Carlos T. Calafate, Pietro Manzoni, and Manuel P. Malumbres. Supporting soft real-time services in MANETs using distributed admission control and IEEE 802.11e technology. In *The 10th IEEE Symposium on Computers and Communications*, June 2005.

[2] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (AODV) routing. Request for Comments 3561, MANET Working Group, http://www.ietf.org/rfc/rfc3561.txt, July 2003. Work in progress.

[3] G-S. Ahn, A. T. Campbell, A. Veres, and L. Sun. Supporting service differentiation for real-time and best effort traffic in stateless wireless ad hoc networks (SWAN). *IEEE Transactions on Mobile Computing*, September 2002.

[4] Hannan Xiao, Kee Chaing Chua, Winston K.G. Seah, and Anthony Lo. On service prioritization in mobile ad-hoc networks. In *IEEE International Conference on Communications (ICC 2001)*, Helsinki, Finland, June 2001.

[5] Hannan Xiao, Winston K.G. Seah, Anthony Lo, and Kee Chaing Chua. A flexible quality of service model for mobile ad-hoc networks. In *IEEE 51st Vehicular Technology Conference Proceedings*, volume 1, pages 445–440, Tokyo, 2000.

[6] IEEE 802.11 WG. IEEE 802.11e/D13.0, "Draft Amendment to Standard for Telecommunications and Information exchange between systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Quality of Service (QoS) Enhancements", April 2005.

[7] K. Fall and K. Varadhan. ns notes and documents. The VINT Project. UC Berkeley, LBL, USC/ISI, and Xerox PARC, February 2000.

[8] Lee Breslau, Ed Knightly, Scott Shenker, Ion Stoica, and Hui Zhan. Endpoint Admission Control: Architectural Issues and Performance. In *Proceedings of ACM Sigcomm 2000*, Stockholm, Sweden, September 2000.

[9] Leonidas Georgiadis, Philippe Jacquet, and Bernard Mans. Bandwidth Reservation in Multihop Wireless Networks: Complexity and Mechanisms. In *International Conference on Distributed Computing Systems Workshops (ICDCSW'04)*, Hachioji - Tokyo, Japan, March 2004.

[10] S.B. Lee, A. Gahng-Seop, X. Zhang, and Andrew T. Campbell. INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks. *Journal of Parallel and Distributed Computing (Academic Press) , Special issue on Wireless and Mobile Computing and Communications*, 60(4):374–406, April 2000.

[11] Sven Wietholter and Christian Hoene. Design and Verification of an IEEE 802.11e EDCF Simulation Model in ns-2.26. Technical Report TKN-03-019, Telecommunication Networks Group, Technische Universitat Berlin, November 2003.